**CHAPTER 1**

**INTRODUCTION**

Motor Insurance Database is a project that allows customers to get insurance quotes using the web application by filling in his/her personal and vehicle details. This project is designed for insurance companies to carry out operations in a smooth and error free manner.

The “Motor Insurance Database” project has been developed to override the complications with the existing manual system. It eliminates the errors that might occur when the quotes are allocated manually and allows you to manage your workforce anytime. The system ensures that every customer is allotted the best policy in the list according to the customer’s needs.

The project uses a MySQL database to store all the related information such as the customers’ personal details, vehicle, policy and insurance claim and settlement details etc. An interactive and easy-to-use front end, developed using HTML, ensures a user friendly and hassle-free experience.

The end-user (customer) can register by entering his/her personal details, followed by the details of the vehicle he/she owns. The use of appropriate triggers minimizes errors while entering data. The customer is allowed to claim for insurance from the insurance company by filling in the online claim form which is validated by the insurance company and it’s either approved or rejected by the insurance company. These claims are arranged in a separate table along with their status of approval. As soon as the claim is approved the customer is given the claimed amount.

This project therefore ensures that customers are allotted their deserving claim amount using certain algorithm the insurance company uses. This project is designed to support the organization in strategic planning and ensures that the organization is equipped with accurate and adequate amount of data to carry out the process of claim allocation.

**CHAPTER 2**

**PROBLEM STATEMENT**

**2.1 OBJECTIVES OF THE PROJECT**

The main objective of the project “Motor Insurance Database” is to manage details of all the registered customers, vehicles and claims. A customer claims for insurance and the PHP code puts all the entered data into the database using a web application. As soon as the amount is approved by the insurance, the concerned customer is allotted the claimed amount and these details are saved.

**2.2 DATABASE ASSUMPTIONS**

* Customer has Cust\_ID, Name consisting of Fname, Mname and Lname, Gender, DOB, Mobile number, Email, License Issue Date , License Issue State, License Number, Address consisting of House no, Street, City and Pin and Activity of the Customer.
* Vehicle has Vehicle ID, Make, Model, Year, Color, Registration Number, Fuel Type, Engine Number, Chassis Number, Date of Purchase and City of Purchase.
* Policy has Policy Number, Issue Date, Effective Date, Expiry Date, Coverage Type, Total Amount and Activity.
* Bill has Bill Number and Date and Amount Paid.
* Payment has Payment ID, Name consisting of First name, Middle Name and Last Name, Bank Name, Account Number, IsCard , Card Number and Expiry Date.
* Insurance has Claim ID, Claim Amount, Damage Type, FIR Number, Date of Claim, Claim Status, Date of Incident and Description of Incident.
* Settlement has Settlement ID, Date and Amount.
* A customer can have any number of vehicles.
* A customer can claim for insurance.
* A customer is therefore allotted with claimed amount.
* Each vehicle has a policy associated with it.
* A policy is associated with more than one bill.
* A bill might have any number of payment methods.

**2.3 REQUIREMENT SPECIFICATION**

**2.3.1. Hardware Requirement**

Processor -INTEL CORE PROCESSOR

Speed - 1.8 GHz

RAM - 1 GB

Hard Disk -1TB

**2.3.2. Software Requirements**

Operating System -WINDOWS10

Programming Language - MySQL, PHP, HTML, CSS

Server -XAMPP

Editor -Notepad ++ , Visual Studio Code

**CHAPTER 3**

**DATABASE DESIGN**

**3.1 BRIEF DESCRIPTION OF RELATIONAL MODEL**

The data and their relationships will be represented by a set of inter-related tables in a relational model. Every table will consist of a column and a row where a column refers to the attribute of an entity and a row refers to the records.

**3.2 ER DIAGRAM**

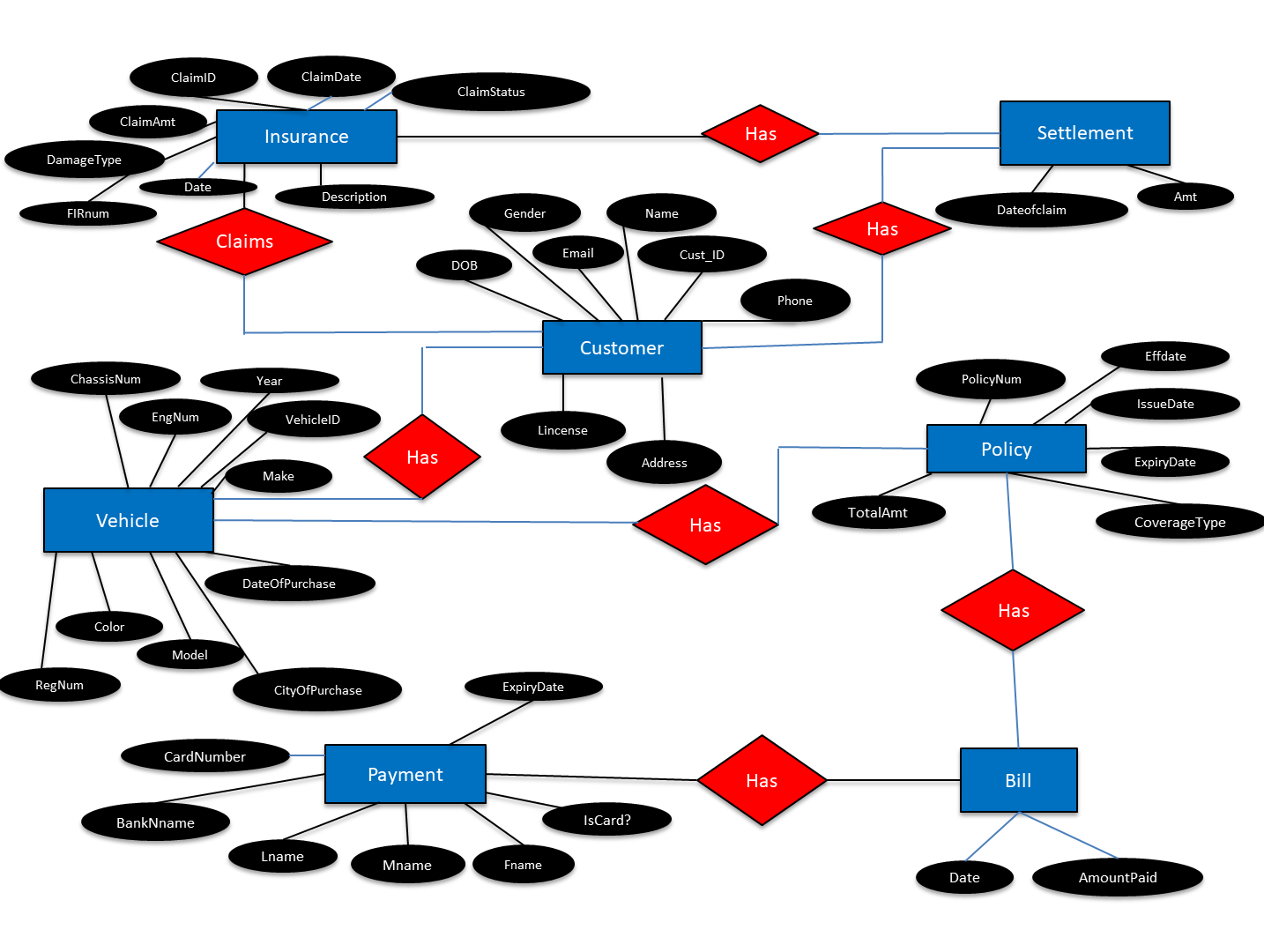
**`**

Fig 3.1

**3.3 RELATIONAL SCHEMA**

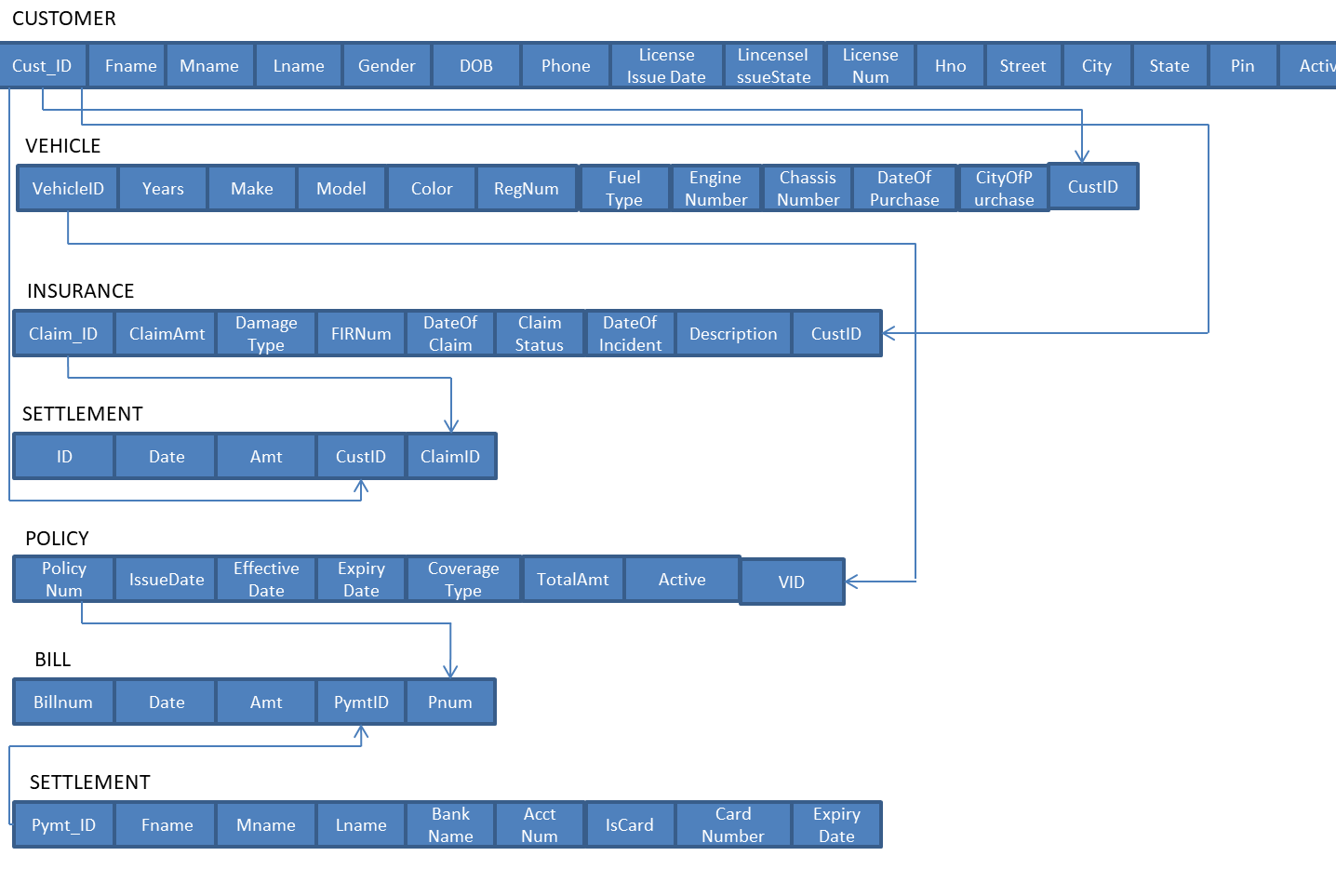
****

Fig 3.2

**CHAPTER 4**

**IMPLEMENTATION**

* 1. **TABLE CREATION**
* CREATE TABLE `payment` ( `Payment\_ID` bigint(20) NOT NULL AUTO\_INCREMENT,`fname` varchar(20) DEFAULT NULL, `Mname` varchar(20) DEFAULT NULL, `Lname` varchar(20) DEFAULT NULL, `BankName` varchar(20) DEFAULT NULL, `AccountNumber` varchar(20) DEFAULT NULL, `IsCard` bit(1) DEFAULT NULL, `CardNumber` bigint(16) DEFAULT NULL, `ExpiryDate` date ,PRIMARY KEY(`Payment\_ID`),UNIQUE KEY `AccountNumber` (`AccountNumber`),UNIQUE KEY `CardNumber` (`CardNumber`));
* CREATE TABLE `settlement` ( `ID` bigint(20) NOT NULL AUTO\_INCREMENT, `Date` date DEFAULT NULL, `Amt` decimal(10,2) DEFAULT NULL, `CustID` bigint(20) DEFAULT NULL, `ClaimID` bigint(20) DEFAULT NULL, PRIMARY KEY (`ID`), KEY `CustID` (`CustID`), KEY `ClaimID` (`ClaimID`), CONSTRAINT `settlement\_ibfk\_1` FOREIGN KEY (`CustID`) REFERENCES `customer` (`cust\_id`) ON DELETE SET NULL ON UPDATE CASCADE, CONSTRAINT `settlement\_ibfk\_2` FOREIGN KEY (`ClaimID`) REFERENCES `insurance` (`claim\_id`) ON DELETE SET NULL ON UPDATE CASCADE);
* CREATE TABLE `bill` (`BillNum` bigint(20) NOT NULL AUTO\_INCREMENT,`Date` date DEFAULT NULL,`AmountPaid` decimal(10,2) DEFAULT NULL,`PaymentID` bigint(20) DEFAULT NULL, `Pnum` varchar(20) DEFAULT NULL,PRIMARY KEY (`BillNum`),KEY `PaymentID` (`PaymentID`),KEY `Pnum` (`Pnum`),CONSTRAINT `bill\_ibfk\_1` FOREIGN KEY (`PaymentID`) REFERENCES `payment` (`Payment\_ID`) ON DELETE SET NULL ON UPDATE CASCADE, CONSTRAINT `bill\_ibfk\_2`

FOREIGN KEY (`Pnum`) REFERENCES `policy` (`PolicyNum`) ON DELETE SET NULL ON UPDATE CASCADE);

* CREATE TABLE `customer` (`Cust\_ID` bigint(20) NOT NULL AUTO\_INCREMENT,`Fname` varchar(20) NOT NULL,`Mname` varchar(20) DEFAULT NULL,`Lname` varchar(20) DEFAULT NULL,`Gender` char(1) DEFAULT NULL,`DOB` date NOT NULL,`Email` varchar(50) DEFAULT NULL,`Phone` bigint(20) NOT NULL,`LincenseIssueDate` date DEFAULT NULL,`LincenseIssueState` varchar(20) DEFAULT NULL,`LincenseNumber` varchar(20) NOT NULL,`Hno` int(11) DEFAULT NULL,`Street` varchar(20) NOT NULL,`City` varchar(20) NOT NULL,`State` varchar(20) DEFAULT NULL,`Pin` varchar(20) DEFAULT NULL,`Active` bit(1) DEFAULT b'0', PRIMARY KEY (`Cust\_ID`),UNIQUE KEY `Phone` (`Phone`), UNIQUE KEY `LincenseNumber` (`LincenseNumber`));
* CREATE TABLE `policy` ( `PolicyNum` varchar(20) NOT NULL,`IssueDate` date DEFAULT NULL,`EffectiveDate` date DEFAULT NULL,`ExpiryDate` date DEFAULT NULL,`CoverageType` varchar(20) DEFAULT NULL,`TotalAmt` decimal(10,2) DEFAULT NULL,`Active` bit(1) DEFAULT b'0',`VID` bigint(20) DEFAULT NULL, PRIMARY KEY (`PolicyNum`),KEY `VID` (`VID`), CONSTRAINT `policy\_ibfk\_1` FOREIGN KEY (`VID`) REFERENCES `vehicle` (`vehicleid`) ON DELETE SET NULL ON UPDATE CASCADE);
* CREATE TABLE `vehicle` ( `VehicleID` bigint(20) NOT NULL AUTO\_INCREMENT,`Year` int(4) DEFAULT NULL,`Make` varchar(20) DEFAULT NULL,`Model` varchar(20) DEFAULT NULL,`Colour` varchar(20) DEFAULT NULL,`RegNum` varchar(10) DEFAULT NULL,`FuelType` varchar(20) DEFAULT 'Petrol',`EngineNumber` int(10) DEFAULT NULL, `ChassisNum` varchar(20) DEFAULT NULL,`CustID` bigint(20) DEFAULT NULL,PRIMARY KEY (`VehicleID`),UNIQUE KEY `RegNum` (`RegNum`), UNIQUE KEY `EngineNumber` (`EngineNumber`),UNIQUE KEY `ChassisNum` (`ChassisNum`), KEY `CustID` (`CustID`),CONSTRAINT `vehicle\_ibfk\_1` FOREIGN KEY (`CustID`) REFERENCES `customer` (`cust\_id`) ON DELETE SET NULL ON UPDATE CASCADE);
* CREATE TABLE `insurance` (`Claim\_ID` bigint(20) NOT NULL AUTO\_INCREMENT,`ClaimAmt` decimal(10,2) DEFAULT NULL, `DamageType` varchar(20) DEFAULT NULL,`FIR\_Number` varchar(20) DEFAULT NULL,`DateofClaim` date DEFAULT NULL,`ClaimStatus` varchar(20) DEFAULT NULL,`CustID` bigint(20) DEFAULT NULL,PRIMARY KEY (`Claim\_ID`),KEY `CustID` (`CustID`),CONSTRAINT `insurance\_ibfk\_1` FOREIGN KEY (`CustID`) REFERENCES `customer` (`cust\_id`) ON DELETE SET NULL ON UPDATE CASCADE);
* INSERT INTO `insurance` VALUES (1,26000.00,'Crash','89372832','2018-10-10','Approved',1);
* INSERT INTO `bill` VALUES (1,'2018-09-03',1370.00,1,'1');

**4.2 QUERIES ON DDL COMMAND**

* ALTER TABLE customer ADD COLUMN age INT;
* ALTER TABLE customer MODIFY age INT DEFAULT 18;
* ALTER TABLE customer DROP COLUMN age;
* ALTER TABLE customer CHANGE Fname name VARCHAR(20);
* ALTER TABLE policy DROP FOREIGN KEY policy\_ibfk\_1;

**4.3 QUERIES ON AGGREGATION FUNCTION**

* SELECT SUM(TotalAmt) FROM policy;
* SELECT COUNT(\*) FROM policy WHERE CoverageType=’Liability’;
* SELECT MAX(Years), Model FROM vehicle;
* SELECT AVG(TotalAmt) FROM policy;

**4.4 QUERIES ON JOINS**

* SELECT Fname, ExpiryDate FROM(policy JOIN customer ON CustID=Cust\_ID) WHERE ExpiryDate>’2019-01-10’;
* SELECT Fname, ExpiryDate FROM (customer LEFT OUTER JOIN policy ON CustID=Cust\_ID);
* SELECT Fname, ExpiryDate FROM(policy RIGHT OUTER JOIN customer ON CustID=Cust\_ID);

**4.5 NESTED AND CORRELATED QUERIES**

* SELECT Fname FROM customer WHERE CustID IN(SELECT CustID FROM vehicle WHERE Make=’TVS’);
* SELECT Fname FROM insurance, customer WHERE CustID IN(SELECT Cust\_ID FROM policy WHERE ClaimStatus=’Approved’;
* SELECT \* FROM customer WHERE(SELECT COUNT(\*) FROM vehicle WHERE CustID=Cust\_ID)>1;
* SELECT Amt FROM settlement X WHERE 2=(SELECT COUNT(DISTINCT Amt) FROM settlement WHERE X.Amt<=Amt);

**4.6 VIEWS**

* CREATE VIEW billdet AS Select Fname AS fname, AmountPaid AS AmountPaid from customer join vehicle join policy join bill where ((CustID = Cust\_ID) and (VID = VehicleID) and (Pnum = PolicyNum));
* CREATE VIEW claimed AS Select Fname AS fname, ClaimAmt AS ClaimAmt from customer join insurance where (CustID = Cust\_ID);
* CREATE VIEW det AS Select Fname AS Fname, RegNum AS RegNum from customer join vehicle where (CustID = Cust\_ID);
* CREATE VIEW policydet AS Select Fname AS fname, PolicyNum AS PolicyNum from customer join vehicle join policy where ((CustID = Cust\_ID) and (VID = VehicleID));
* CREATE VIEW sett AS Select Fname AS fname, ID AS ID from customer join settlement where (CustID = Cust\_ID);

**4.7 TRIGGERS**

* DELIMITER //

CREATE TRIGGER t1

BEFORE INSERT ON customer

FOR EACH ROW

BEGIN

IF ((CURRENT\_DATE-new.DOB)/10000)<18 THEN

SIGNAL SQLSTATE '45000' SET

MESSAGE\_TEXT ='DOJ should be greater than 18';

END IF;

END; //

DELIMITER;

* DELIMITER //

CREATE TRIGGER t2

BEFORE INSERT ON POLICY

FOR EACH ROW

BEGIN

IF ((new.ExpiryDate-new.EffectiveDate/10000))<1 THEN

SIGNAL SQLSTATE '45000' SET

MESSAGE\_TEXT ='Minimum length of the policy in 1 year';

END IF;

END; //

DELIMITER;

**CHAPTER 5**

**SAMPLE OUTPUT**

**5.1 Using MySQL**

1. SELECT \* FROM bill;

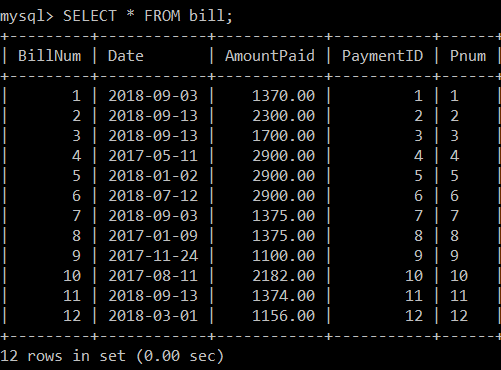


Fig.5.1

1. SELECT \* FROM policy WHERE CoverageType=’Liability’;

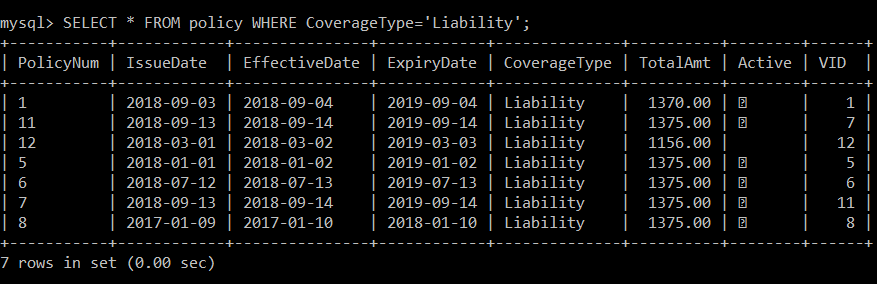


Fig.5.2

1. SELECT COUNT(\*) AS COUNT, Colour FROM vehicle GROUP BY Colour HAVING COUNT(\*)>2;

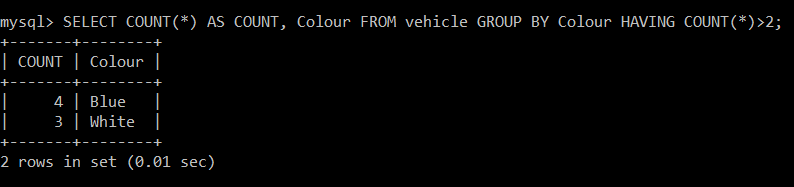


Fig.5.3

1. SELECT RegNum,ExpiryDate FROM(policy JOIN vehicle ON VID=VehicleID) WHERE ExpiryDate>'2019-01-01';

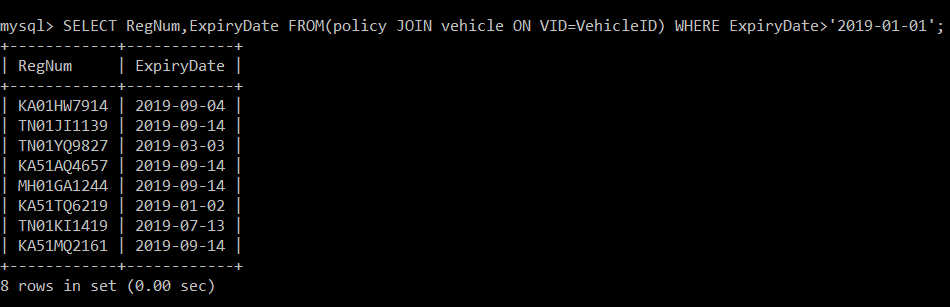


Fig.5.4

**5.2 Using HTML and PHP**

1. Homepage

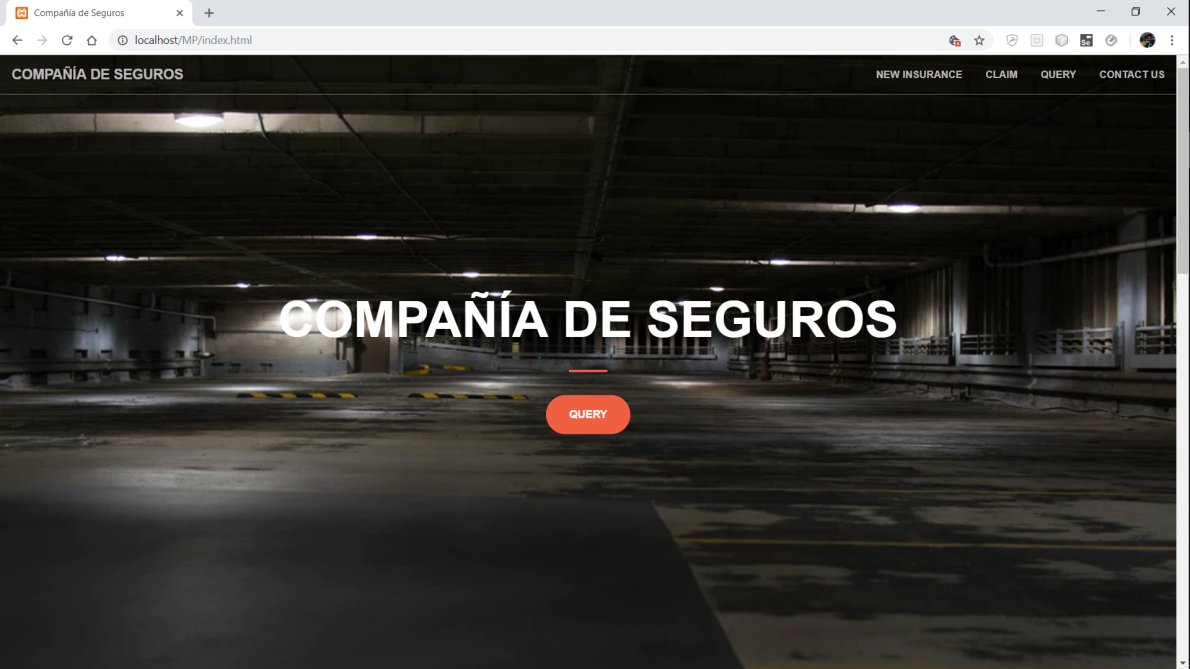


Fig 5.5

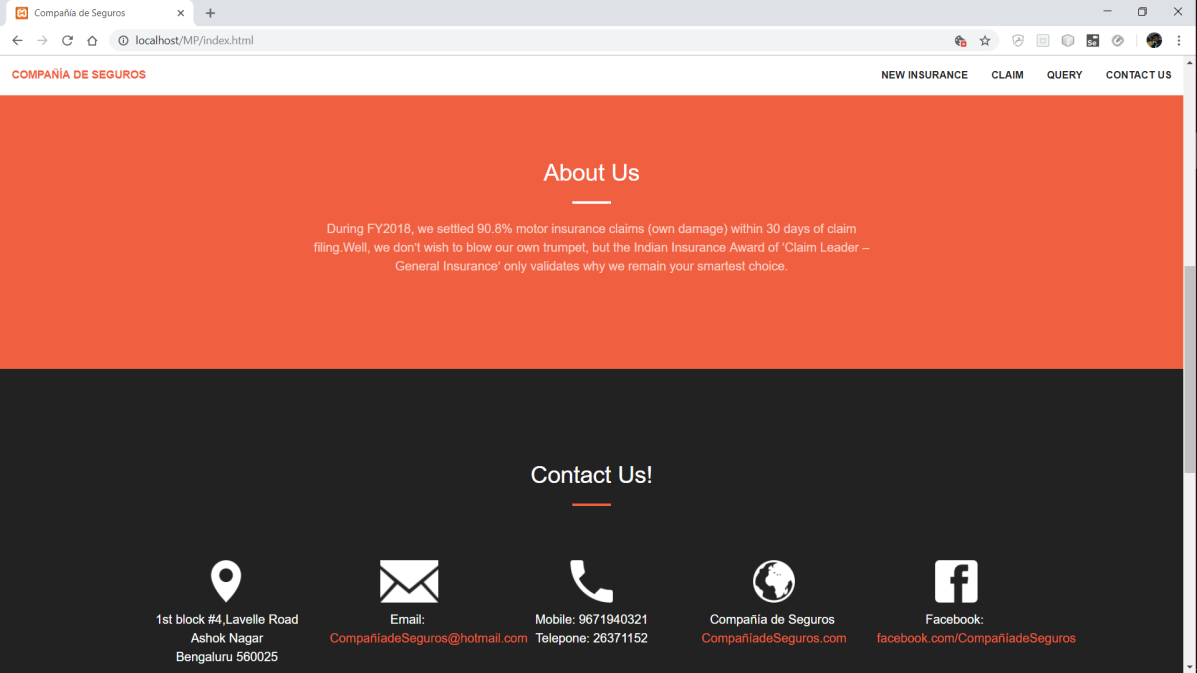


Fig 5.6

1. Vehicle details form

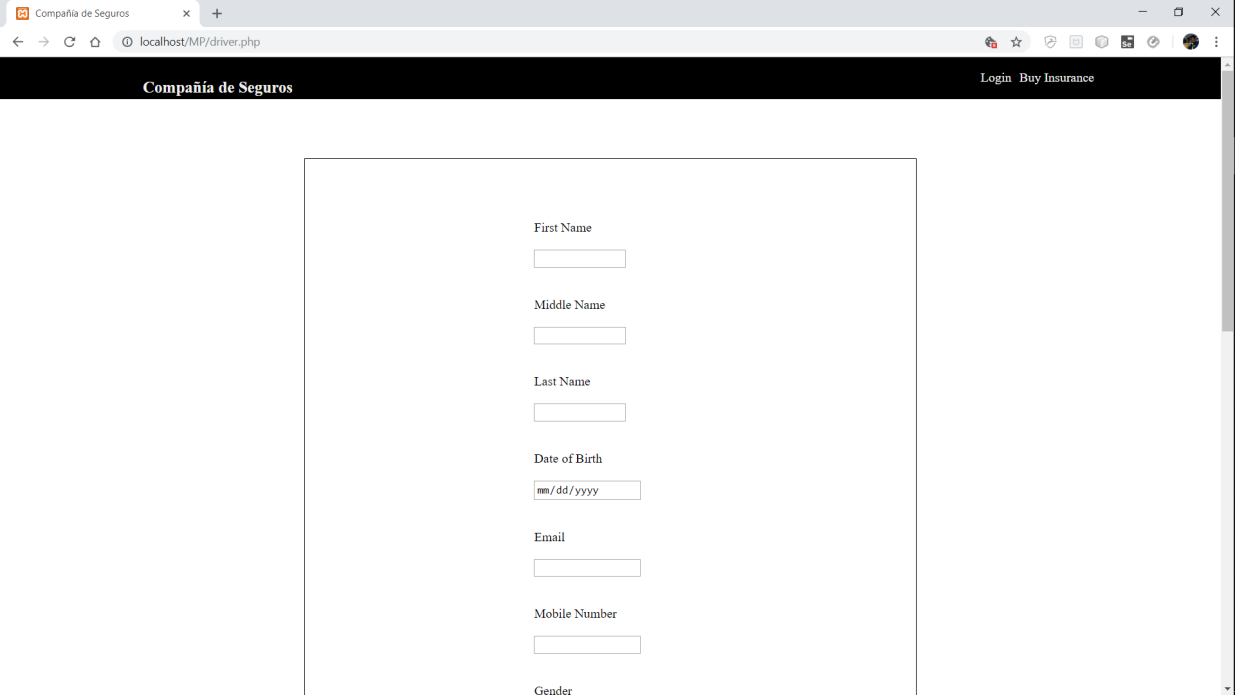


Fig 5.7

1. Customer details form

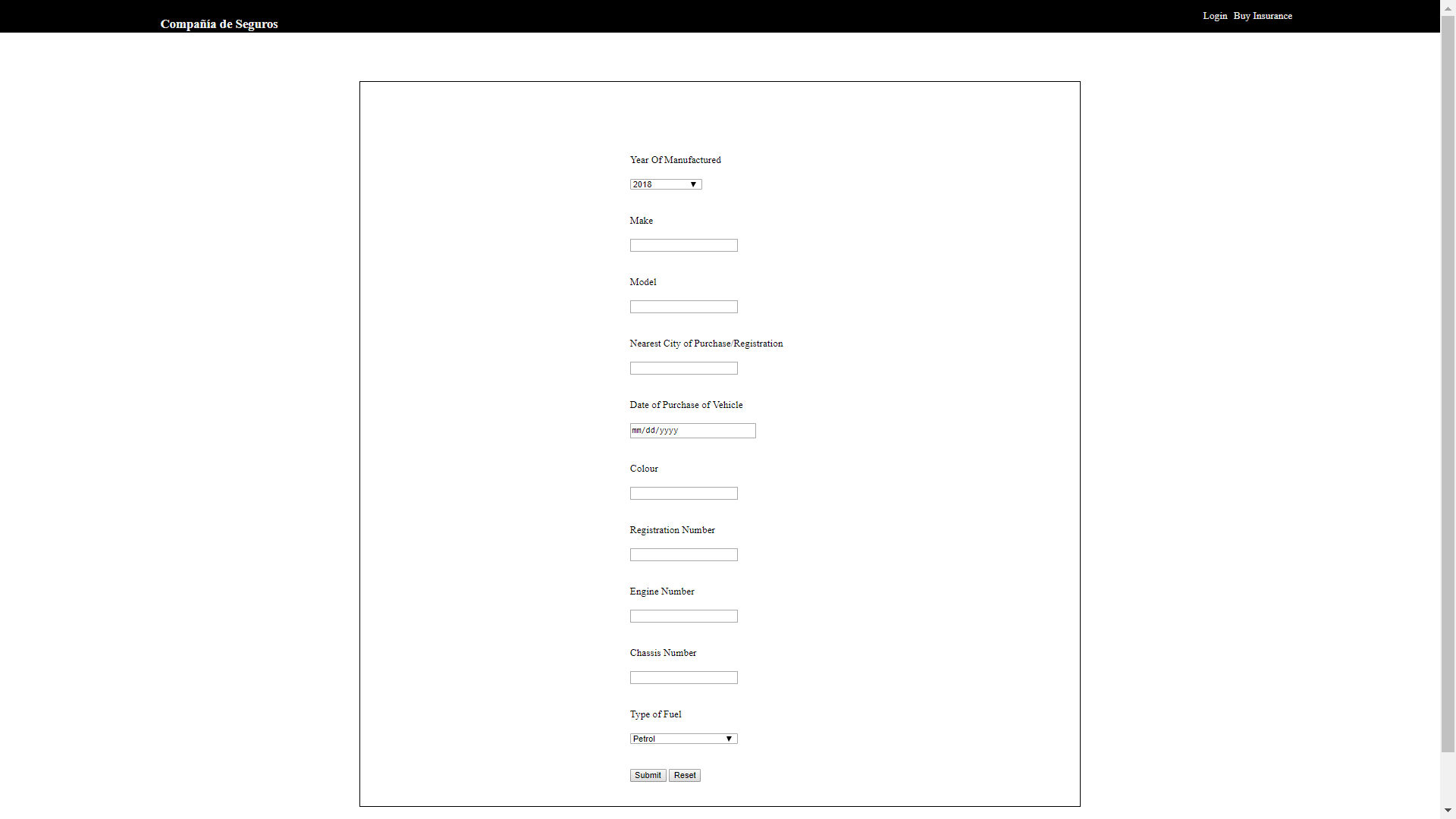


Fig 5.8

1. Claim details form

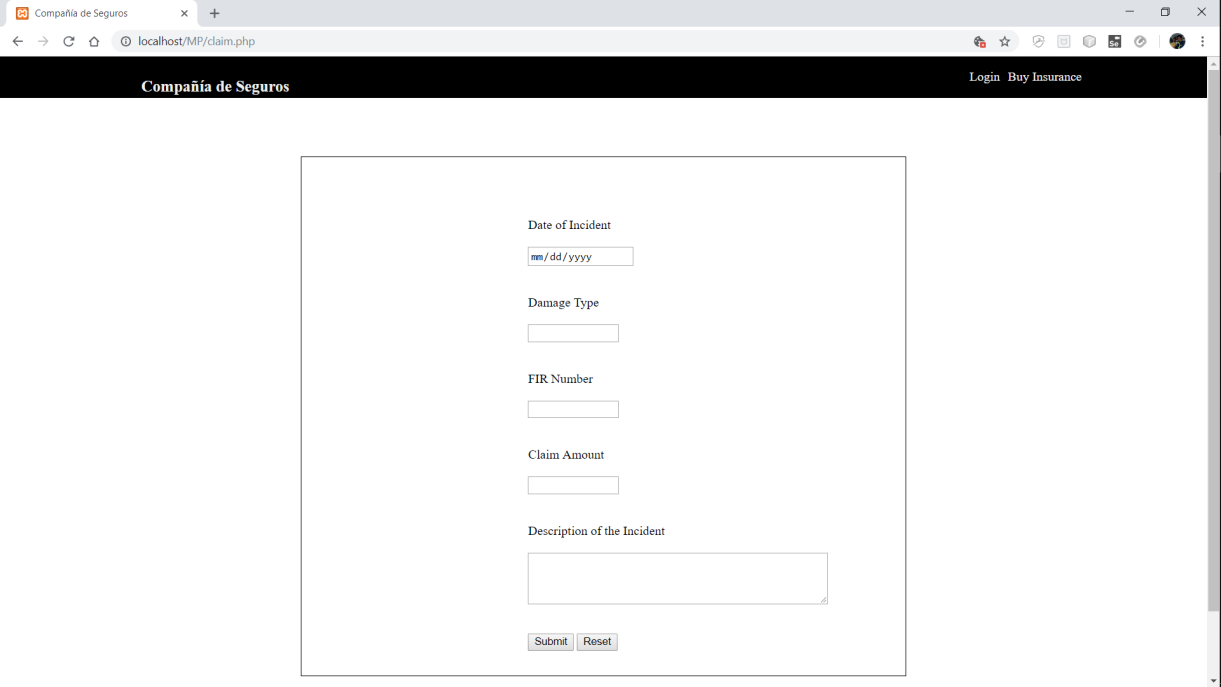


Fig 5.9

1. Select Query page

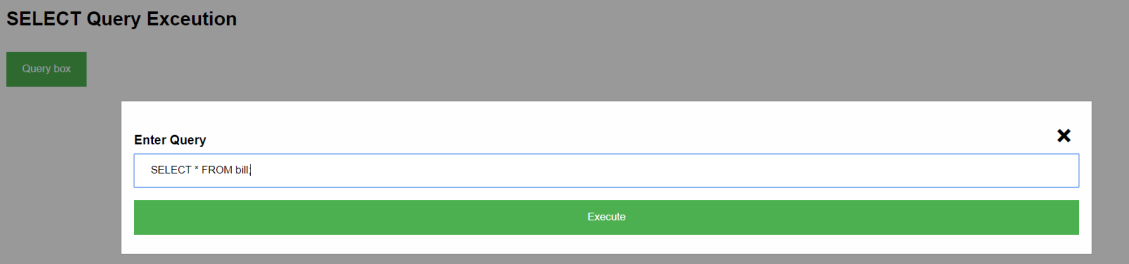


Fig 5.10

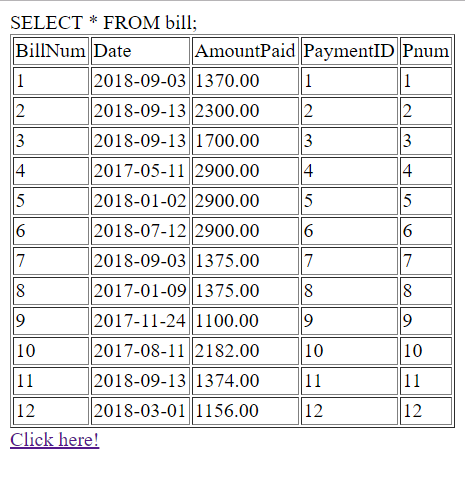


Fig 5.11

1. DML Query page

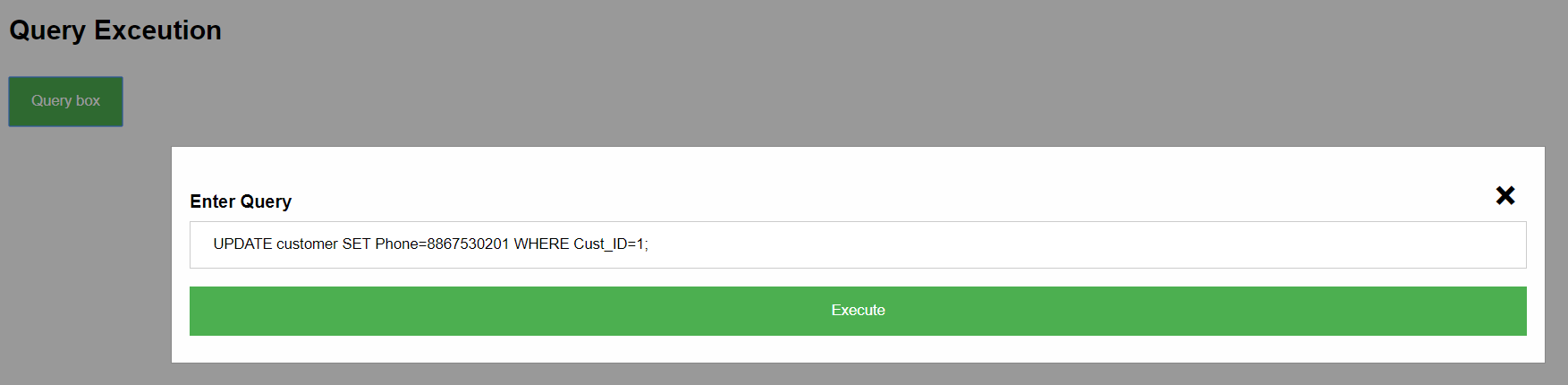


Fig 5.12

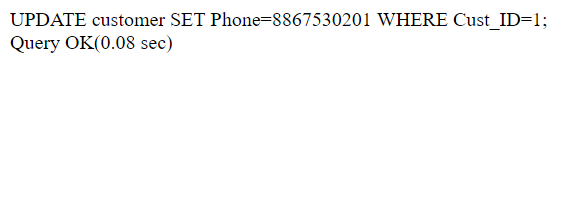


Fig 5.13

**CHAPTER 6**

**CONCLUSION**

This project aims at reducing the burden on the insurance companies to maintain records of the customer and vehicles along with the claim details. It is simple and easy to use, allowing the user to enter his/her details, and his claim details in case he/she wants to claim insurance on a user friendly and error free environment. User data is updated and checked for authenticity using appropriate triggers. The project ensures that the registered customers are able to claim for insurance by filling in the claim insurance form on the web application. This prevents any sort of anomaly or ambiguity, thus making it highly efficient and competent.

**CHAPTER 7**

**BIBLIOGRAPHY**

* [www.scribd.com](http://www.scribd.com)
* [www.bainbridge.co.in](http://www.bainbridge.co.in)
* [www.nevonprojects.com](http://www.nevonprojects.com)
* [www.kashipara.com](http://www.kashipara.com)